**Отчёт**

Для реализации алгоритма «Проверка тождественной истинности ДНФ» были созданы следующие классы (рисунок 1):

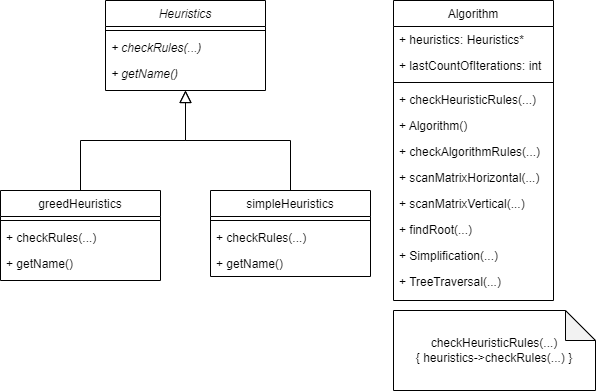


Рисунок 1 – диаграмма реализованных классов

Метод findRoot класса Algorithm осуществляет поиск корня уравнения. Метод принимает на вход объект класса BoolEquation и возвращает приведенный к строке интервал, являющийся решением. Если решение не существует, выводится предупреждение и возвращается вектор, состоящий из неопределенных компонент.

В качестве правил, позволяющих сократить ранг корня уравнения, были использованы две эвристики:

1. greedHeuristics (использовалась при решении домашней работы).

В ситуации, когда ни одно из правил алгоритма не выполняется, выбираем строку с минимальным рангом и рассматриваем определенные компоненты выбранной строки, подсчитываем, сколько нулей и единиц в каждом столбце и выбираем ту компоненту, в столбце которой наибольшее количество ячеек с одинаковыми значениями.

1. simpleHeuristics (использовалась на лекции при разборе примера).

Выбираем наиболее определенный столбец и присваиваем компоненте значение, противоположное значению большинства компонент в данном столбце.

Эвристики сравнивались по количеству итераций, подсчет которых велся по мере сокращения множества интервалов. greedHeuristics продемонстрировала результат хуже по сравнению с simpleHeuristics на всех примерах (за исключением тех, где для подтверждения отсутствия корня потребовался перебор всех альтернатив).

Также при поиске корня были использованы объекты классов Interval, BBV, NodeBoolTree и BoolEquation. Классы Interval и BBV совпадают с предложенными в лекциях. Классы NodeBoolTree и BoolEquation были изменены.

Класс NodeBoolTree содержит следующие член-данные:

* unsigned int var – номер переменной xi разложения.
* int count – мощность множества интервалов;
* Interval\*\* dnf – множество интервалов;
* BBV varsToBeConsidered – маска на столбцы: 1 столбец подлежит рассмотрению, 0 – не рассматривается;
* BBV rowsToBeConsidered – аналогичная маска на строки;
* bool fixVal\_0 – флаг, указывающий на фиксацию значения 0 для рассматриваемой перемемнной xi (var);
* bool fixVal\_1 – флаг, указывающий на фиксацию значения 1 для рассматриваемой перемемнной xi (var);
* Interval Solution – для сохранения решения.

По мере поиска решения соответствующие значения компонент фиксируются в Solution, а при «удалении» строк и столбцов матрицы по правилам алгоритма соответствующие компоненты масок varsToBeConsidered и rowsToBeConsidered помечаются нулями.

Член-данные класса BoolEquation:

* Interval\*\* setIntevals – множество интервалов;
* int num\_rows – количество строк;
* int count\_var – количество переменных.

Класс BoolEquation содержит методы readFromPla и isSolution. Первый заполняет матрицу данными из pla-файла. Второй принимает на вход интервал и проверяет, является ли он решением уравнения, заданного данным множеством интервалов.

Кроме того, при поиске решения используется вспомогательная структура данных Component, которая в качестве член-данных содержит номер компоненты и её значение, выбранные на текущей итерации алгоритма.

В функции main демонстрируется работа алгоритма на примерах. Программа выполняется в несколько шагов. В качестве примеров были использованы предложенные примеры (DNF\_Random\_examples) и несколько примеров, составленных самостоятельно. Директория "DNF\_Random\_examples/my\_examples/" содержит два тривиальных примера my\_pla\_4.pla и my\_pla\_5.pla, которые нужны для демонстрации работы алгоритма, когда уравнение не имеет корней.

Сначала из примеров отбираются pla-файлы, являющимися корректными (для проверки корректности используется функция isPlaFileCorrect).

Для каждого отобранного примера находим корень (или убеждаемся, что его не существует). Сначала ищем корень с помощью greedHeuristics, затем с помощью simpleHeuristics. Помимо найденного корня выводится количество итераций, потребовавшееся для нахождения корня. После нахождения корня идет проверка найденного решения на то, что это действительно корень, и выводится сообщение о результате проверки.

На рисунках 2 и 3 продемонстрированы результаты для файлов dnfRnd\_8.pla и my\_pla\_5.pla.

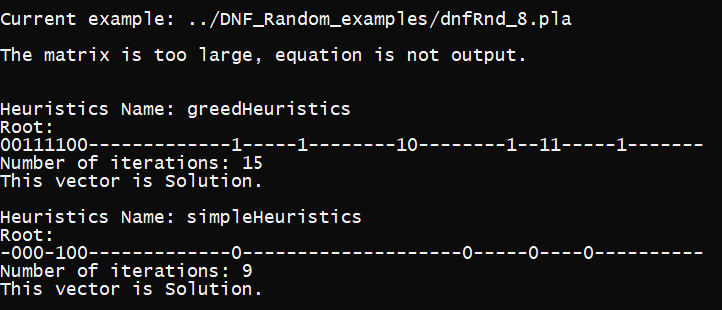


Рисунок 2 – dnfRnd\_8.pla

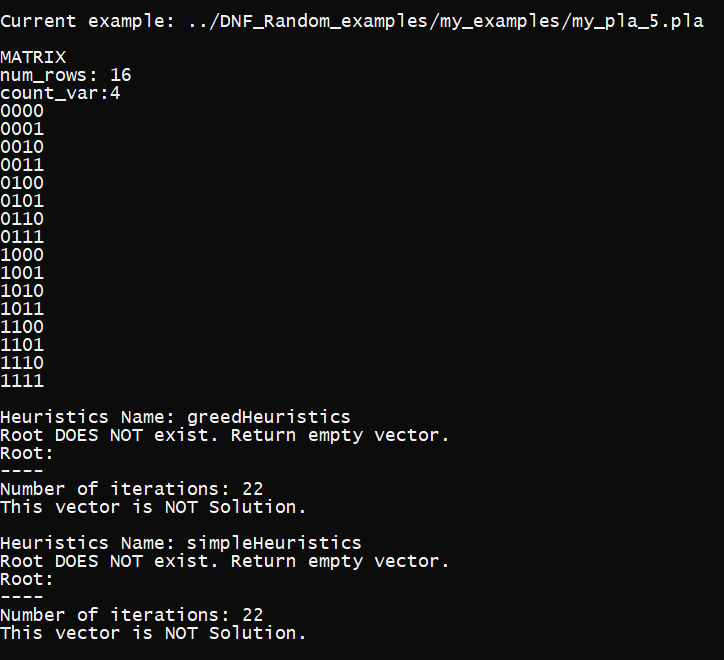


Рисунок 3 – my\_pla\_5.pla